

THE INVENTION CLAIMED IS:

1. A remote procedure call optimizing method for
optimizing remote procedure calls (called RPCs hereunder)
5 between a server object (called a server hereunder)
offering at least one remote procedure on the one hand, and
a client object (called a client hereunder) carrying out
processing by use of an RPC calling said at least one
remote procedure on the other hand on a computer for
10 executing at least either one program or one program part,
said remote procedure call optimizing method being
characterized by the steps of:

adding to said server a new remote procedure
integrating a plurality of RPCs constituting a processing
15 part of said client into a smaller number of RPCs to be
executed, as well as an interface of said new remote
procedure; and

allowing said client to call the newly added remote
procedure to carry out said processing part comprising said
20 smaller number of RPCs.

2. A remote procedure call optimizing method for
optimizing RPCs between a server offering at least one
remote procedure on the one hand, and a client carrying out
processing by use of an RPC calling said at least one
25 remote procedure on the other hand on a computer for

00405089 092799
662260 68050460

executing at least either one program or one program part,
said remote procedure call optimizing method being
characterized by the steps of:

analyzing a source code of said client so as to
5 detect a remote procedure execution sequence constituting a
set of RPCs that are highly likely to be executed
successively;

adding to said server a new remote procedure for
executing said remote procedure execution sequence; and

10 allowing said client to call the newly added remote
procedure to carry out said processing part comprising said
smaller number of RPCs.

3. A remote procedure call optimizing method for
optimizing RPCs between a server offering at least one
15 remote procedure, and a client carrying out processing by
use of an RPC calling said at least one remote procedure,
on a computer for executing at least either one program or
one program part, when an IDL description for said at
least one remote procedure is provided, said remote
20 procedure call optimizing method being characterized by the
steps of:

analyzing a source code of said client so as to
detect a remote procedure execution sequence constituting
an array of RPCs that are highly likely to be executed
25 successively;

662260" 68050460

determining a new remote procedure for executing
said remote procedure execution sequence in a single RPC;
and

5 additionally storing an interface of the determined
remote procedure into said IDL description.

4. A remote procedure call optimizing method for
optimizing RPCs between a server offering at least one
remote procedure, and a client carrying out processing by
use of an RPC calling said at least one remote procedure,
10 on a computer for executing at least either one program or
one program part, when an IDL description for said at least
one remote procedure is provided, said remote procedure
call optimizing method being characterized by the steps of:

15 inputting a source code and an IDL description of
said client;

generating a first remote procedure comprising a
plurality of RPCs included in the input client source code,
and an IDL description of said first remote procedure; and

20 outputting a source code of the generated first
remote procedure, said IDL description of said generated
first remote procedure, and a new client source code
including a call to said generated first remote procedure.

5. A remote procedure call optimizing method for
optimizing RPCs between a server offering at least one
25 remote, and a client carrying out processing by use of an

00405089 092799
00/260" 68050460

RPC calling said at least one remote procedure, on a computer for executing at least either one program or one program part, when an IDL description for said at least one remote procedure is provided, said remote procedure call
5 optimizing method being characterized by the steps of:

inputting a source code and an IDL description of said client, and a source code of said server;

generating a first remote procedure comprising a plurality of RPCs included in the input client source code,
10 and an IDL description of said first remote procedure; and

outputting a source code of the generated first remote procedure, said IDL description of said generated first remote procedure, and a new client source code including a call to said generated first remote procedure.

15 6. A remote procedure call optimizing method for optimizing RPCs between a server offering at least one remote procedure, and a client carrying out processing by use of an RPC calling said at least one remote procedure, on a computer for executing at least either one program or
20 one program part, said remote procedure call optimizing method being characterized by the steps of:

inputting a source code of said client and an object of said server;

generating a first remote procedure comprising a
25 plurality of RPCs included in the input client source code;

00405089 092799 002260 68050460

and

outputting a source code of the generated first remote procedure, and a new client source code including a call to said generated first remote procedure.

5 7. A remote procedure call optimizing method for optimizing RPCs between a server offering at least one remote procedure, and a client carrying out processing by use of an RPC calling said at least one remote procedure, on a computer for executing at least either one program or
10 one program part, said remote procedure call optimizing method being characterized by the steps of:

inputting a source code of said client and a source code of said server;

generating a first remote procedure comprising a
15 plurality of RPCs included in the input client source code; and

outputting a source code of the generated first remote procedure, and a new client source code including a call to said generated first remote procedure.

20 8. A remote procedure call optimizing method for optimizing RPCs between a server offering at least one remote procedure, and a client carrying out processing by use of an RPC calling said at least one remote procedure, on a computer for executing at least either one program or
25 one program part, when an IDL description for said at least

00405080 092709 002260

one remote procedure is provided;

wherein said IDL description includes a syntactic structure declaring either presence or absence of side effects of said at least one remote procedure,

5 interchangeability of a plurality of remote procedures in execution order thereof, and feasibility of parallel execution of a plurality of remote procedures;

said remote procedure call optimizing method being characterized by the steps of:

10 inputting said IDL description and a source code of said client;

analyzing flows of data in the client source code to detect an RPC string that may be integrated; and

15 adding an interface of said RPC string to said IDL description.

9. A remote procedure call optimizing method according to any one of claims 3, 4, 5, 6, 7 and 8, further comprising the step of acquiring said IDL description by:

20 (1) reading an IDL source file storing said IDL description;

(2) communicating with said server;

(3) referring to a file of said server; or

(4) communicating with an interface repository providing said IDL description.

25 10. A remote procedure call optimizing method

00405089 092799
662260 68050460

according to any one of claims 4 through 7, further comprising the steps of:

(1) detecting from said client source code a processing part including at least two RPCs issued in close proximity to one another;

(2) calculating a first procedure causing said server to carry out a process equivalent to said processing part;

(3) replacing said processing part of said client source code with a call to said first procedure, thereby generating a new client source code;

(4) adding an interface of said first procedure to said IDL description; and

(5) adding a definition of said first procedure to either the added server source code or said new server source code.

11. A remote procedure call optimizing method according to claim 10, wherein said detecting step comprises the step of searching through said client source code for a basic block in which at least two RPCs are issued, thereby detecting a processing part ranging from a first RPC to a second RPC within said basic block.

12. A remote procedure call optimizing method according to claim 10, wherein said detecting step comprises the step of detecting as said processing part a loop including a basic block having at least one RPC.

13. A remote procedure call optimizing method

09405089.092799
06/26/00 08:05:04

according to claim 10, wherein said step of calculating said first procedure comprises the step of acquiring input and output arguments of said first procedure using data flow analysis.

5 14. A remote procedure call optimizing method according to claim 10, wherein said step of calculating said first procedure comprises the step of judging whether an alias state will occur in which the same data are either referenced or changed by a plurality of variable names
10 within said processing part.

 15. A remote procedure call optimizing method according to claim 13 or 14, wherein said step of judging alias state occurrence comprises the step of making a judgment using data flow analysis in terms of variable
15 types.

 16. A remote procedure call optimizing method according to claim 13 or 14, further comprising the step of supplementing said new client source code, where said alias state can occur, with a branching process whereby said
20 processing part is carried out if an alias exists and said procedure is called if no alias exists.

 17. A remote procedure call optimizing method according to claim 10, wherein said step of calculating said first procedure comprising the steps of: estimating
25 transfer data quantities of input and output arguments

00405089-092700

necessary for a call to either said first procedure or a second procedure; and stopping the replacing of said processing part with said call to said first procedure if the estimated transfer data quantities exceed a
5 predetermined level.

18. A remote procedure call optimizing method according to claim 17, wherein said estimating step comprises the steps of: using a communication measuring part of said client or of said server to measure the number
10 of RPCs, an input argument transfer data quantity, and an output argument transfer data quantity at program run time; and estimating said transfer data quantity based on run time communication information measured by said communication measuring part.

19. A remote procedure call optimizing method according to claim 10, wherein said detecting step (1) comprises the step of changing an execution order of RPCs depending on the presence or absence of side effects or on the interchangeability of a plurality of remote procedures
20 in execution order thereof as stated in claim 8, thereby detecting a processing part including at least two RPCs issued in close proximity to one another.

20. A remote procedure call optimizing method according to claim 10, wherein said step of calculating
25 said first procedure in (2) comprises the step of

09405069-092799
002260-68050460

generating in said first procedure a source code for
executing a plurality of parallelly executable remote
procedures in a plurality of parallel execution units
called threads depending on the feasibility of parallel
5 execution as stated in claim 8.

21. A remote procedure call optimizing method
according to any one of claims 1 through 8, wherein said
server includes an extensible dispatcher for accepting new
remote procedures that are added, said extensible
10 dispatcher apportioning calls to said at least one remote
procedure.

22. A remote procedure call optimizing method
according to any one of claims 1 through 21, wherein said
server incorporates either an interpreter or an execution
15 environment for interpreting and executing a language, and
wherein said client sends to said server a script in which
said client executes a plurality of RPCs, said server
having said script analyzed and executed by either said
interpreter or said execution environment.

20 23. A program execution method combining at least
two of:

a program for executing a remote procedure call
optimizing method according to any one of claims 1 through
8;

25 an IDL compiler for converting an IDL description

00405009-092700

into a source code for carrying out RPCs of a server and a client; and

a compiler for converting said source code to an executable code.

5 24. A CORBA IDL compiler, a Sun RPC stub generator, a Java IDL compiler, or a Java RMI compiler incorporating a program for executing a remote procedure call optimizing method according to any one of claims 1 through 8.

10 25. A program execution method for utilizing a program for executing a remote procedure call optimizing method according to any one of claims 1 through 8, as a preprocessing program for:

15 a CORBA IDL compiler or a Sun RPC stub generator; or at least one of a Java IDL compiler and a Java RMI compiler.

26. A storage medium accommodating a computer program for causing a computer to carry out a program for executing a remote procedure call optimizing method according to any one of claims 1 through 8.

20 27. A program execution method comprising the steps of:

 inputting a client source code and an IDL source code of each client to a program for executing a remote procedure call optimizing method according to any one of
25 claims 1 through 8;

00405089-002790

outputting from said program a new client source
code and a new IDL source code of each client as well as an
additional server source code of each server in
correspondence with a remote procedure call optimized by
5 said program;

inputting said new IDL source code to an IDL
compiler which in turn outputs a client stub, an RPC header
file and a server stub;

662260" 68050460
10 compiling and linking said client stub, said RPC
header file and said server stub, as well as said new
client source code of each client and said additional
server source code of each server, thereby outputting a
client object and an additional server object;

15 dynamically linking said additional server object
and a server object to output a new server object;

executing said client object on a first computer;

transmitting an RPC from said client object in order
to execute said client object on said first computer; and

20 executing on a second computer said new server
object having received said RPC.

28. A program execution method according to claim 27,
wherein said transmitting step comprises the step of
transmitting RPCs from a first communication measuring part
of said first computer to a second communication measuring
25 part of said second computer, said first and said second

communication measuring part measuring run time
communication information.

29. A program execution method according to claim 28,
wherein said transmitting step comprises the step of
5 transmitting run time communication information measured by
at least one of said first and said second communication
measuring part, to said program for executing said remote
procedure call optimizing method.

30. A program execution method comprising the steps
10 of:

inputting a client source code and an IDL source
code of each client and a server source code of each server
to a program for executing a remote procedure call
optimizing method according to any one of claims 1 through
15 8;

outputting from said program a new client source
code and a new IDL source code of each client as well as a
new server source code of each server in correspondence
with a remote procedure call optimized by said program;

20 inputting said new IDL source code to an IDL
compiler which in turn outputs a client stub, an RPC header
file and a server stub; and

compiling and linking said client stub, said RPC
header file and said server stub, as well as said new
25 client source code of each client and said new server

00405060" 092799

source code of each server, thereby outputting a client object and a server object.

31. A program execution method comprising the steps of:

5 inputting a client source code of each client and a server class code of each server to a program for executing a remote procedure call optimizing method according to any one of claims 1 through 8;

10 outputting from said program a new client source code of each client and an additional server source code of each server in correspondence with a remote procedure call optimized by said program;

15 compiling the output new client source code of each client and the output additional server source code of each server so as to output a client class code and an additional server class code which may be executed by a Java virtual machine (called JVM) each;

20 compiling the output additional server class code using a remote method invocation (called RMI) compiler in Java so as to output a client stub and a server stub;

 processing the output client class code and the output client stub using a first JVM; and

 executing the output additional server class code and the output server stub using a second JVM.

25 32. A program execution method comprising the steps

09405089.092799

of:

inputting a client source code of each client and a server source code of each server to a program for executing a remote procedure call optimizing method

5 according to any one of claims 1 through 8;

outputting from said program a new client source code of each client and a new server source code of each server in correspondence with a remote procedure call optimized by said program;

10 compiling the output new client source code of each client and the output new server source code of each server so as to output a client class code and a server class code which may be executed by a Java virtual machine (called JVM) each;

15 compiling an output additional server class code using a remote method invocation (called RMI) compiler in Java so as to output a client stub and a server stub;

processing the output client class code and the output client stub using a first JVM; and

20 executing the output additional server class code and the output server stub using a second JVM.

*add
A4*

09405089-092799